

FSE PROGRAM SUMMARY DOCUMENT #5

Transfer Function Routine XFRRUN

R. E. McFarland

November 1990

NASA

National Aeronautics and
Space Administration

Document: FSE PROGRAM SUMMARY #5, orig. Feb. 15, 1990, final Nov. 1, 1990

Title: Transfer Function Routine XFRRUN

Author: R. E. McFarland, NASA

Function: Real-time solution to arbitrary-order ratios of Laplace polynomials using various discrete realization algorithms.

Availability: (1) SIMDEV VAX, \$DISK 1:[STRIKE]XFRRUN.FOR
(2) FSD VAX, \$LIBRARY:[STRIKE]XFRSET.FOR
(3) ADDEV VAX, \$DISK 1:[MCFARLAND.STRIKE]XFRSET.FOR
(4) Author, PC Diskette

Language: FORTRAN, Computer Portable, ANSI Standard

References: (1) Transfer Function Routine XFRSET, FSE Program Summary Document #4, R. E. McFarland, Feb. 1990.

(2) On Optimizing Computations for Transition Matrices, R. E. McFarland, IEEE Trans. Automatic Control, Vol. AC-23, No. 3, June 1978.

(3) Digital Signal Processing, Alan V. Oppenheim and Ronald W. Schaffer, Prentice-Hall, Inc., 1975.

(4) State Variables for Engineers, Paul M. DeRusso, Rob J. Roy and Charles M. Close, Wiley & Sons, Inc., New York, 1967.

(5) Analyzing Delays in a Flight Simulation Environment, R. E. McFarland and J. W. Bunnell, AIAA-90-3174, Flight Simulation Technologies Conference, Sept. 17-19, 1990, Dayton, Ohio.

Summary

Transfer functions are usually presented in Laplace form, and in order to simulate them on a digital computer system a "discrete realization technique" is required. Many such techniques exist, but none of them produce the exact characteristics of the original continuum formulation. Response differences occur because of the assumptions, and because of the finite cycle time "T", where certain techniques degrade faster than others as a function of the cycle time, and display different characteristics over the discrete bandwidth (limited by the Nyquist criterion). Of all the available techniques, *state space* techniques are superior. For example, a stable transfer function always produces a stable discrete realization using state space techniques, independent of the cycle time.

At the Ames flight simulation laboratory state space techniques are utilized fully. This means that Laplace functions are combined whenever possible, in order to create transfer functions of the highest order. The results are invariably better than those obtained using piecemeal integration and differentiation operators. This combination process is limited only because of "embedded nonlinearities."

Unlike other techniques, a state space formulation requires a "data hold." This is the

pure form of modeling; the Laplace transform becomes the "pulse transform" that actually represents data in a digital computer, *i.e.*, only a series of pulses is available, one at each and every "T". The programming engineer must merely make a decision about the behavior of the data between the sampling instants (guided by knowledge about the characteristics of the particular input data). If values are indeed assumed to vanish between sampling instants then the z-transform of the pulse transform is the function that should be directly coded. This assumption is generally not very useful in flight simulation, however. More realistic data hold assumptions are discussed herein. Additionally, the "data hold" will be seen to influence the proper time of output applicability. "Advancing" and "concurrent" forms are discussed.

This document is the second in a series of three: XFRSET, XFRRUN and XFRBOD. Together, these three documents describe a discrete realization process for the solution to arbitrary ratios of polynomial transfer functions, using state space techniques. The first document, FSE Program Summary Document #4, XFRSET, explains how the transition matrix,

$$\begin{bmatrix} \Gamma \end{bmatrix} = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} & \cdots & \gamma_{1N} \\ \gamma_{21} & \gamma_{22} & \gamma_{23} & \cdots & \gamma_{2N} \\ \gamma_{31} & \gamma_{32} & \gamma_{33} & \cdots & \gamma_{3N} \\ \vdots & \vdots & \vdots & & \vdots \\ \gamma_{N1} & \gamma_{N2} & \gamma_{N3} & \cdots & \gamma_{NN} \end{bmatrix}$$

and forcing function vector

$$\begin{bmatrix} \Lambda \end{bmatrix} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_N \\ \lambda_{N+1} \end{bmatrix}$$

are created, and placed in a buffer SETB. This buffer is a function of the denominator coefficients "a_n" and the cycle time "T". The document (see ref. 1) also previews how this buffer can be used for the transition of the states of the system.

The transition process is best performed by subroutine XFRRUN, the subject of this paper. In addition, subroutine XFRRUN handles the linear combination of numerator coefficients for more general ratios of polynomial transfer functions. These transfer functions are assumed to have the form,

$$\frac{y(s)}{u(s)} = \frac{b_{M+1}s^M + b_M s^{M-1} + \dots + b_2 s + b_1}{s^N + a_N s^{N-1} + \dots + a_2 s + a_1}$$

where $M \leq N$. In ref. 1 (XFRSET) it is shown how this Nth order system is equivalent

to the state variable equations for a linear system given by,

$$\begin{aligned}\dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{U}(t) \\ \mathbf{y}(t) &= \mathbf{B}\mathbf{x}(t)\end{aligned}$$

where \mathbf{A} is the essential matrix of the system expressed in canonical form,

$$[\mathbf{A}] = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ -a_1 & -a_2 & -a_3 & \dots & -a_N \end{bmatrix}$$

the input is a scalar,

$$\mathbf{U}(t) = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ u(t) \end{bmatrix}$$

and the \mathbf{B} matrix couples the state variables to the output.

The above matrix equations are a state space representation of a single-input N^{th} order linear differential equation. Such a system arises naturally from the Laplace description of a typical transfer function defined by the ratio of polynomials, when $M < N$. For the case of $M = N$ it will be shown that solutions are also available, although they involve approximations when "advancing holds" are used.

As an extension to the foundation document of ref. 2, the "XFR" software includes options for three different data hold assumptions: (1) the zero-order data hold, in which the output is *purportedly* advanced one cycle from the input; (2) the first-order data hold, in which the output is (also) advanced by one cycle; and (3) the triangular data hold, where the output is concurrent with the input. For this last case the system may also include $M = N$ without the necessity of using an approximation.

XFRRUN propagates the state variables depending upon the selected data hold. Although XFRRUN is primarily intended for the solution of transfer functions using only state space techniques, an additional option is also available. XFRRUN will also produce the bilinear transform (Tustin) solution to an arbitrary-order transfer function. This algebraic substitution technique is discussed in Appendix A, and is included in the software and documentation for completeness. Hopefully, users will make comparisons

between the triangular data hold and the bilinear technique and discover that triangular hold solutions are as good, and in some cases vastly superior to solutions using the bilinear transform (*e.g.*, as in the case of a notch filter). This is because the bilinear transform manifests "frequency warping", as discussed in ref. 3.

The third program in this series, XFRBOD, produces gain and phase data for any of the selected options, and also establishes baseline comparison values from the original Laplace transfer functions.

Calling Sequence

CALL XFRRUN (IMODE, IHOLD, T, N, A, M, B, SETB, RUNB, U, Y)

The calling sequence for this program contains the following quantities:

IMODE	Conventional mode value (as in BASIC common, where this common is not used in the XFR-software).
IHOLD	The selected data hold: IHOLD = 0 designates the zero-order data hold, advancing the output by one cycle from the input (for step inputs). IHOLD = 1 designates the first-order data hold, advancing the output by one cycle from the input. IHOLD = 2 designates the triangular data hold, with output concurrent with the input. IHOLD = 3 designates the bilinear algorithm, with output concurrent with the input. Calls to XFRSET are not required using this option.
T	The cycle time, in seconds.
N	The order of the denominator s-plane polynomial ($1 \leq N \leq 20$ is the current software dimension limitation).
A()	A vector dimensioned N or greater, consisting of the denominator coefficients a_n . A(N+1) is not required; it is always assumed unity. (It usually takes N + 1 coefficients to express an N th order polynomial).
M	The order of the numerator s-plane polynomial. Usually $M < N$, but equal order systems $M = N$ permitted if IHOLD = 2. (For $M = N$ the advancing holds must use an approximation, which should be avoided).
B()	A vector dimensioned M + 1 or greater, consisting of the numerator coefficients b_m . B(M+1) is not assumed to have a specific value (unlike A(N+1)), and therefore must be included in the vector.

SETB()	A buffer unique to the denominator of dimension of at least $N^2 + N + 1$. This buffer is computed by XFRSET, and used by XFRRUN.
RUNB()	A buffer unique to the input and state variables. This buffer must be dimensioned at least $N + 1$. As shown below, this buffer is used for a repository of the system states (first N cells) and the previous value of the input (the last, <i>i.e.</i> , $N+1^{\text{st}}$ cell).
U	The scalar input.
Y	The scalar output.

Setup Function

A call to subroutine XFRSET is required to obtain the buffer SETB. XFRSET, as explained in ref. 1, is called initially and, for nonstationary denominator coefficients, may be called periodically, or whenever these coefficients change "significantly." Although "significantly" is not a well defined term in this context, the slowly-varying coefficient hypothesis would have it that coefficient frequency content be much lower than the input/output frequency content. Also, engineering judgement dictates that discrete changes in the location of the poles should not excite the transfer function. Hence, for nonstationary transfer functions some care should be exercised in the selection of the period of calling XFRSET.

If the call to XFRSET must occur every cycle, then it is doubtful that the transfer function itself is amenable to a Laplace representation.

If only the bilinear option is to be used (IHOLD = 3), a state space solution is not produced; hence, a call to XFRSET is not required.

Run Function

XFRRUN is called in all modes (IMODE) in simulation models. Its calling sequence contains "U", the scalar input, and "Y", the scalar output. In I.C. mode (IMODE < 0) the output is the input times the transfer function's D.C. gain, if any, and zero otherwise.

The states are themselves available to the calling program in the buffer RUNB. Thus, the default I.C. option may be overridden, and initial conditions may be imposed on all integrators in the system.

XFRRUN's procedures vary with IHOLD, as is shown below. An understanding of these procedures is required for the proper use of the software. This involves a consideration of the concept of "state." Also, users must understand the difference between an "advancing" data hold and a "concurrent" data hold.

"The state of a discrete time system can be intuitively defined as the minimum amount of information about the system which is necessary to determine both the output and the future states of the system, if the input function is known." (Ref. 4). In this document the states of a system are defined as the outputs of integrators, where "x" is used to designate the state vector, " x_n " is used to define an individual (spatial) element in the

state vector, and " $x_{n,k}$ " is used to define the n^{th} element at the discrete time $t = kT$ (" k " is referred to as the "temporal index"). This state vector definition reduces the expression for the scalar output " $y_{(k)}$ " to a simple linear combination of states and numerator coefficients.

For simulation models at Ames, the *outputs* of any aero, control or engine module should be concurrent with the beginning of the cycle. This is because these outputs are generally proportional to forces or moments and, at the end of the cycle, subroutine STRIKE (or SMART) combines these module outputs, applicable at t_k , and integrates them to velocities and positions, which are then applicable at t_{k+1} , or the beginning of the next interval (or end of the current interval). Since the computer workload " T " has accrued during this interval, these values are then immediately applicable for output and data storage, with the time stamp t_{k+1} . (Note, however, that the inputs and accelerations are all applicable at time stamp t_k).

Hence, *advances in module outputs, wherein transfer functions usually reside, are erroneous*. Gross errors in the time subscript, sometimes called "temporal index" problems, produce phase errors of up to 180° (at the Nyquist frequency).

Temporal index problems arise when a transfer function is inadequately prepared for discrete realization. Although advances in *total* module outputs are almost invariably undesirable, this does not mean that advancing options are not useful. Indeed, *internal* to various modules they are often required, especially in feedback paths. The feedback path that closes on the pilot, however, is completely handled by the Ames kinematic structure (STRIKE or SMART). Fig. 1 demonstrates this fact.

The three basic data hold options available in XFRRUN are (1) IHOLD = 0, the zero-order data hold, which advances the output by one cycle, (2) IHOLD = 1, the first-order data hold, which also advances the output by one cycle, and (3) IHOLD = 2, the triangular data hold, which delivers an output that is concurrent with the input. Whenever possible, the triangular hold should be used. The bilinear transform technique, available when IHOLD = 3, should be avoided.

Within XFRRUN the state vector x is created each computer cycle, and is placed in the first N cells of the buffer RUNB. The "spatial index" is " n " ($n = 1, 2, \dots, N$), and x_n is used to designate the $(n-1)^{\text{st}}$ derivative of the primary state x_1 . Since this discussion concerns discrete systems, the temporal index (k) designates the time point of applicability ($t = kT$),

$$x_k = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}_k$$

The individual elements of this state vector, including up to the $(N-1)^{\text{st}}$ derivative for the N^{th} order polynomial, may be expressed by differentiation with respect to time:

$$(x_n)_k = \frac{d^{n-1}x(t)}{dt^{n-1}} \quad (t = kT), \quad (n = 1, 2, \dots, N), \quad (k = 0, 1, 2, \dots)$$

The cell RUNB(N+1) is used for storing the previous input u_{k-1} .

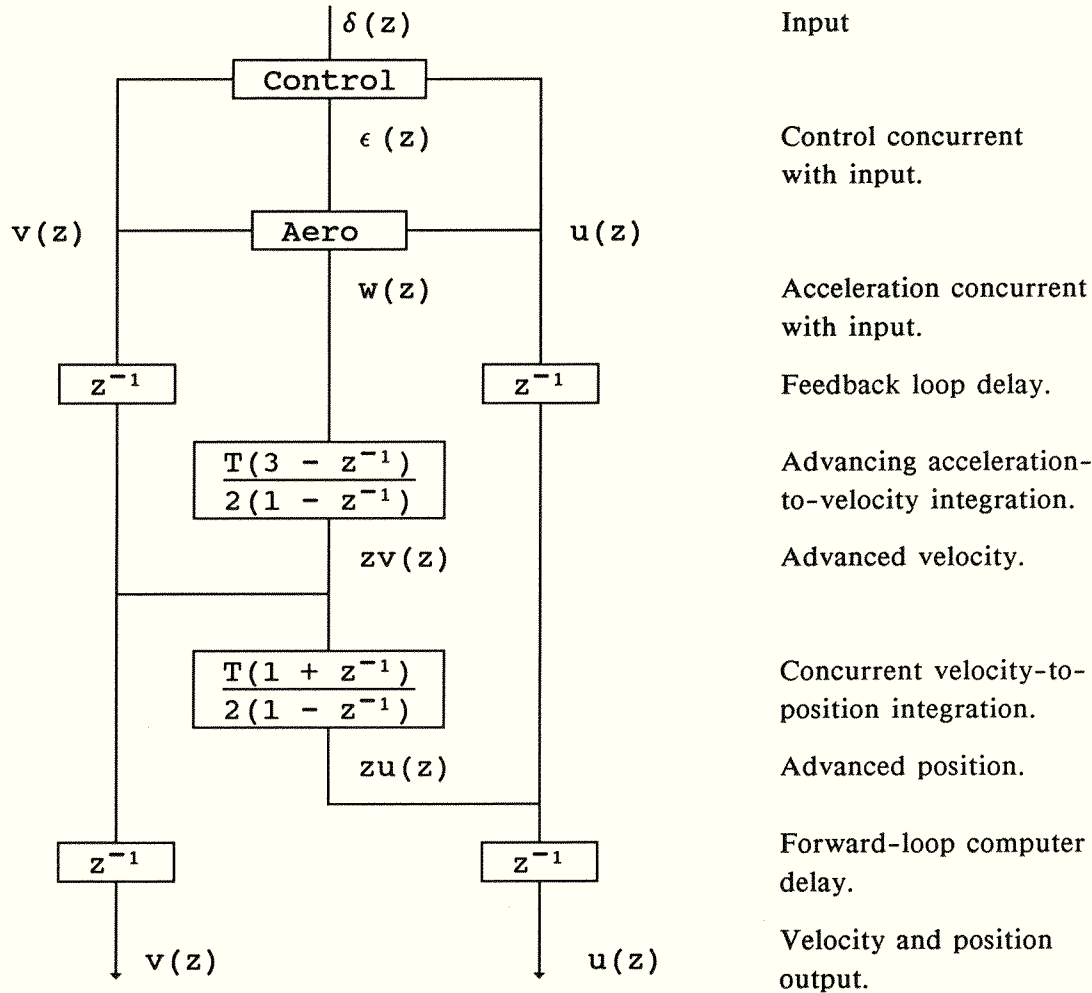


Fig. 1 – Ames Integration Flow Structure

Data Holds

The buffer SETB is available to XFRRUN once XFRSET has been called. The three data hold options within XFRRUN provide for three different state-space methods of transition. The first two (IHOLD = 0, 1) advance the output one cycle from the input, and the third (IHOLD = 2) produces a concurrent output. The equations that update the states are given as follows:

IHOLD = 0 (Zero Order Hold, Advancing)

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}_{k+1} = \begin{bmatrix} \Gamma \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}_k + \begin{bmatrix} \lambda_2 \\ \lambda_3 \\ \vdots \\ \lambda_{N+1} \end{bmatrix} u_k$$

If the transfer function was simply integration $f(s) = 1/s$, this IHOLD would produce the Euler integration algorithm (advanced output):

$$y_{k+1} = y_k + T u_k$$

(For integration examples the numerator is unity; hence $(y_k = x_k)$).

The "concurrent form" of a z-transform is here defined as explicitly showing the advancing functionality, if any. In creating block diagrams, for example, this is very useful because computer delays and feedback paths are then also included in the graphics (see ref. 5).

Procedure

A procedure for developing the concurrent form of a difference equation and the correct z-transform equivalent to the difference equation is as follows: Shift the indices such that the input's highest index is "k", designated as "current", and let the output subscripts take their appropriately-shifted index values. This produces a realizable system (real and nonanticipative) providing that the resultant highest index of the output is "k" or higher. If this is the case the system is realizable (and hence capable of being simulated in real time). Then, solve for the highest-indexed output value, where the order of the index displays the number of shifts. This establishes "k" as the input point (referencing the "time of the input"), and if the highest index of the output is also "k", the transfer function does not produce a time shift. Indeed, a time shift becomes immediately obvious using this technique, and should be retained in the z-transform description of the process, because a direct comparison may then be made with a continuum functionality $f(s)$. In the process of creating a z-transform equivalent of the difference equation, variables with the subscript "k" are assigned the power $z^0 = 1$ in the z-transform representation. For example, in the case of Euler integration we write the shifted output as,

$$\frac{zy(z)}{u(z)} = \frac{T}{1 - z^{-1}}$$

which may be directly compared to $f(s) = 1/s$ (at the input point). This defines out "concurrent form", clearly displaying the shifted output in this case. The phase of this function relative to integration $1/s$ produces a *lead* of $T/2$. Then, if the end-of-cycle performance is desired, the computer delay z^{-1} is used to multiply this expression. This then produces the well-known *delay* of $T/2$ resultant from Euler integration.

IHOLD = 1 (First Order Hold, Advancing)

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}_{k+1} = \begin{bmatrix} \Gamma \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}_k + \begin{bmatrix} \lambda_2 \\ \lambda_3 \\ \vdots \\ \lambda_{N+1} \end{bmatrix} u_k + \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_N \end{bmatrix} \left[\frac{u_k - u_{k-1}}{T} \right]$$

If the transfer function was simply integration $f(s) = 1/s$, IHOLD = 1 would produce the Adams second order integration algorithm (advanced output):

$$y_{k+1} = y_k + \frac{1}{2}T(3u_k - u_{k-1})$$

This produces the Adams second-order integration algorithm in the form of a concurrent z-transform (with explicit shifts),

$$\frac{zy(z)}{u(z)} = \frac{\frac{1}{2}T(3 - z^{-1})}{1 - z^{-1}}$$

where the phase lead is shown (use of concurrent z-transforms means that block diagrams then show computer delays as they occur).

A discussion on concurrent forms may be found in ref. 5. Also, the documentation on subroutine XFRBOD will further explore difference equations and their equivalent z-transform representations.

IHOLD = 2 (Triangular Hold, Concurrent)

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}_k = \begin{bmatrix} \Gamma \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}_{k-1} + \begin{bmatrix} \lambda_2 \\ \lambda_3 \\ \vdots \\ \lambda_{N+1} \end{bmatrix} u_{k-1} + \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_N \end{bmatrix} \left[\frac{u_k - u_{k-1}}{T} \right]$$

If the transfer function was simply integration $f(s) = 1/s$, IHOLD = 2 would produce the triangular integration algorithm, also known as trapezoidal integration.

$$y_k = y_{k-1} + \frac{1}{2}T(u_k + u_{k-1})$$

The triangular hold algorithm is generally superior to all others when the input functionality is assumed to consist of representative samples of continuous data. An "advancing form" of the triangular hold is not possible.

IHOLD = 3 (Bilinear Transform, Concurrent)

The bilinear transform (Tustin) is not a state-space technique. It is interesting to note that for single integration, and single integration only, the bilinear transform produces the same result as the triangular algorithm. Indeed, this fact partially accounts for its undeserved popularity (the main reason is undoubtedly its simplicity). The bilinear transform, which should *not* be preferred over the triangular data hold, also does not permit an "advancing form." A generalized form of the bilinear transform is discussed in Appendix A.

Numerator Combinations

XFRRUN performs two distinct jobs. The first job, involving only the denominator coefficients, is the propagation of the state vector "x". This job is complicated by the different algorithms available for the various data holds. The second job is the linear combination of states with the numerator coefficients, producing the scalar output "y". This job is complicated in the case of equal order systems ($M = N$).

Considering the linear combination of states required to compute the scalar output "y", $M + 1$ states must be available. For $M < N$, all $M + 1$ states are indeed available from the above XFRRUN procedures, regardless of whether a concurrent or advancing hold is used. Hence, using the numerator coefficients b_m , for a concurrent data hold the output is,

$$Y = y_k = \sum_{m=1}^{M+1} b_m x_{m,k}$$

and for an advancing hold the output is,

$$Y = y_{k+1} = \sum_{m=1}^{M+1} b_m x_{m,k+1}$$

The concurrent triangular hold solution ($IHOLD = 2$) also permits a solution for equal-order systems ($M = N$) because of the relationship,

$$x_{N+1,k} = u_k - \sum_{n=1}^N a_n x_{n,k}$$

which is just a restatement of the original (denominator) transfer function. This equation therefore provides the missing state derivative when $M = N$.

However, a problem arises when $M = N$ using the advancing holds. This problem is easily seen by shifting all of the "k" temporal subscripts in the above equation to "k+1". The derivative of the N^{th} state (required when $M = N$) may only be obtained for advancing options by approximating u_{k+1} , the *future* value of the input (anticipatory)!

The software of XFRRUN does allow $M = N$ in the case of advancing holds, but the user is cautioned that the approximation $u_{k+1} \approx u_k$ is used. The Bode plot software of XFRBOD will clearly show the errors produced by using this approximation.

The buffer RUNB is used to store the previous value of the state vector (x), and the previous value of the input (u). Hence, it must have the dimensions of $N + 1$. This buffer must be unique for each transfer function call regardless of whether the denominator coefficients (and hence SETB) are the same. This is because these quantities depend upon (1) the previous and current inputs, (2) the selected data hold, and (3) the previous states.

Examples

First Example

In the first example we have a control system module with input U and output Y, as acted upon by the transfer function given by,

$$f(s) = \frac{s}{(s + r_1)(s + r_2)}$$

Since the output of this system (in the Ames simulation environment) eventually will produce a force or moment, the concurrent hold IHOLD = 2 is used. The pertinent code is,

```

      DIMENSION A(2),B(2),SETB(7),RUNB(3)
      DATA R1/2/,R2/3/,B/0,1/,N/2/,M/1/,T/0.02/
C
C ONE SHOT:
      A(1) = R1*R2
      A(2) = R1 + R2
      CALL XFRSET(T,N,A,NTERM,SETB)
C
C ALL MODES:
      CALL XFRRUN(IMODE,2,T,N,A,M,B,SETB,RUNB,U,Y)

```

Second Example

In our second example we assume that the transfer function has a feedback path, with nonstationary gain "G". Later this functionality will be combined with the original transfer function, as it should be whenever possible. This form is shown here because it is illustrative of what must be done if the feedback path contains a limiter.

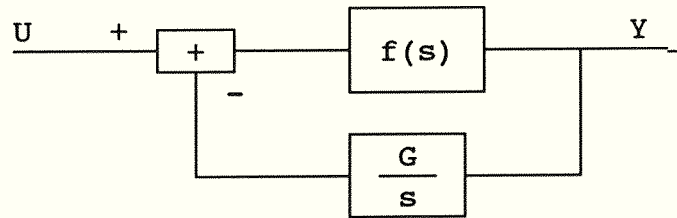


Fig. 2 - System With Feedback

Expanding from the first example, we only need one extra buffer, and one extra call to XFRRUN to developed the advanced output (IHOLD = 1):

```

      DIMENSION RUNF(3)
C
C ALL MODES:
      UM = U - YF
      CALL XFRRUN(IMODE,2,T,N,A,M,B,SETB,RUNB,UM,Y)
      CALL XFRRUN(IMODE,1,T,N,A,0,G,SETB,RUNF,UM,YF)

```

"Y" is the output, and "YF" is the advanced output, as required in the input equation. Note that only one SETB buffer is required in this contrived example; the transfer functions may be written with identical denominators by changing only the numerators.

Third Example

For a third example, consider coefficient variations in real time. Since "G" is a numerator gain, the above example does not require further effort. However, suppose the denominator coefficients change with, say, altitude. Then, since XFRSET need only be called periodically, we would have something like the following complete system:

```

      DIMENSION A(2),B(2),SETB(7),RUNB(3)
      DIMENSION RUNF(3)
      DATA N/2/,M/1/,K/10/,B/0,1/
C
C ALL MODES:
      K = K + 1
      IF(K.GE.10) THEN
        K = 1
        A(1) = R1*R2
        A(2) = R1 + R2
        CALL XFRSET(T,N,A,NTERM,SETB)
      END IF
C
      UM = U - YF
      CALL XFRRUN(IMODE,2,T,N,A,M,B,SETB,RUNB,UM,Y)
      CALL XFRRUN(IMODE,1,T,N,A,0,G,SETB,RUNF,UM,YF)

```

Fourth Example

For the fourth and final example (but see Appendix B), we note that despite the fact that nonstationary coefficients may be present in the system, the transfer function should be represented in a form that takes advantage of the state space techniques. This produces the new function,

$$f(s) = \frac{s}{(s + r_1)(s + r_2) + G}$$

such that the entire system is handled by,

```

      DIMENSION A(2),B(2),SETB(7),RUNB(3)
      DATA N/2/,M/1/,K/10/,B/0,1/
C
C ALL MODES:
      K = K + 1
      IF(K.GE.10) THEN
          K = 1
          A(1) = R1*R2 + G
          A(2) = R1 + R2
          CALL XFRSET(T,N,A,NTerm,SETB)
      END IF
C
      CALL XFRRUN(IMODE,2,T,N,A,M,B,SETB,RUNB,U,Y)
```

Concluding Remarks

The XFR- software constitutes a very powerful tool for the real-time solution to arbitrary ratios of transfer functions.

The third document in the "XFR" series deals with Bode data. The gain and phase of a discrete representation are useful for comparison with the original continuum transfer function, and for isolating coding errors when using XFRRUN.

For simulation models at Ames, the outputs of any aero, control or engine module should be concurrent with the beginning of the cycle (*i.e.*, use IHOLD = 2), since the integrations in STRIKE (or SMART) require linear and angular accelerations that are applicable at the beginning of the interval.

If $M = N$ the advancing holds (IHOLD = 0 or 1) use the approximation $u_{k+1} = u_k$. This could produce completely erroneous results. However, the concurrent triangular hold (IHOLD = 2) is exact.

The IHOLD = 3 option should be avoided unless you are quite confident that its application will not significantly distort results. For the case of notch filters it must be avoided because of frequency warping, *i.e.*, the notch will not be where you want it to be. This distortion will also occur with more simple systems, where the distortion, for

example, of the cutoff frequency, will be a function of the ratio of the cutoff frequency to the Nyquist frequency.

The XFR- software replaces the following software:

- (1) FACT and UPDATE
- (2) TACT
- (3) SOLUS
- (4) SFILTER
 - (a) INTEG
 - (b) FOLO
 - (c) FOHI
 - (d) SOLO
 - (e) NSFOLO

Appendix A

The Bilinear Transform

The bilinear transform is also known as the Tustin algorithm. By writing the N^{th} order differential equation as a set of N first-order equations, this technique is seen to consist of integrating each of these differential equations by use of numerical approximations to integrals. This method should be avoided unless distortion in the frequency axis can be tolerated or compensated. For instance, the bilinear transform will not accurately simulate a notch filter without frequency prewarping (which requires extensive massage in gain/phase space). Even lowpass filters with linear phase characteristics cannot be obtained by applying the bilinear transformation to analog lowpass filters with linear phase characteristics (Ref. 4, p. 211).

The technique reduces to an algebraic substitution for "s" wherever it appears in the Laplace form of the transfer function,

$$s = \frac{2(z - 1)}{T(z + 1)}$$

where "z" is the z-transform operator that may easily be used to develop difference equations. Defining $x_{n,k}$ as the n^{th} state at time $t = kT$, the bilinear transform produces the states from state derivatives using repeated trapezoidal integrations ($n = 1, 2, \dots, N$),

$$x_{n,k} = x_{n,k-1} + \frac{1}{2}T(x_{n+1,k} + x_{n+1,k-1})$$

except that the (starting) derivative of the N^{th} state is not a state. It is given by,

$$x_{N+1,k} = u_k - \sum_{n=1}^N a_n x_{n,k}$$

and is itself a function of the integrals. This does not appear to be very useful because the states are determined from their derivatives. However, using induction on these processes, a convolution occurs which permits expressing the highest derivative in terms of the current input and only previous states:

$$x_{N+1,k} = \frac{u_k - \sum_{j=1}^N \left\{ \left(\frac{1}{2}T \right)^{j-1} \sum_{n=j}^N a_{n+1-j} \left[x_{n,k-1} + \left(\frac{1}{2}T \right) x_{n+1,k-1} \right] \right\}}{1 + \sum_{j=1}^N \left(\frac{1}{2}T \right)^j a_{N+1-j}}$$

Hence, for a polynomial system of any order, the concurrent, highest derivative may be determined, and it is a sufficient kernel for the development of all of the states. This occurs because of the repeated integration structure of the bilinear transform.

As in the case of the triangular hold formulation, the numerator terms of the transfer function may be included by a simple linear combinations of available terms.

$$Y_k = \sum_{m=1}^{M+1} b_m X_{m,k}$$

The numerator order may thus be as large as the denominator order ($M = N$).

Note that, similar to the preferable triangular data hold algorithm, an "advancing form" is not available. Also, both of these algorithms will be up to 180° out of phase if they are used in feedback paths.

The bilinear transform method is an available option in both subroutine XFRRUN, and subroutine XFRBOD. For this option, which is selected by IHOLD = 3, a call to the setup routine XFRSET is not required (using this option the buffer SETB is ignored).

Using the bilinear transform algorithm, both nonstationary numerator and denominator coefficients are permitted by XFRRUN.

Appendix B

Double Integration

Examples using the various IHOLD options are developed here. The complete function in these examples is given by the impulse response to the transfer function,

$$e(s) = \frac{1}{s[(s + \alpha)^2 + \beta^2]}$$

and by consulting any table of Laplace transfer functions the "baseline" output is given for all time ($t \geq 0$) by,

$$y(t) = \left\{ 1 - e^{-\alpha t} [(\alpha/\beta)\sin(\beta t) + \cos(\beta t)] \right\} / (\alpha^2 + \beta^2)$$

The sample problem is separated into components here for illustrative purposes. In order to separate the transform into a time-varying input and a discrete modeling portion that consists of double integration, the original function is separated into the two Laplace functions,

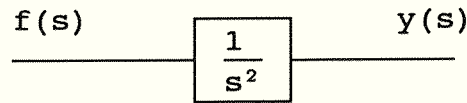


Fig. 3 - Separated Problem

where $f(s) = s^2 e(s)$ such that,

$$f(s) = \frac{s}{(s + \alpha)^2 + \beta^2}$$

The driving function $f(t)$ may then be produced from Laplace tables:

$$f(t) = e^{-\alpha t} [\cos(\beta t) - (\alpha/\beta)\sin(\beta t)]$$

Hence, the separated problem may be shown as the sample-data system,

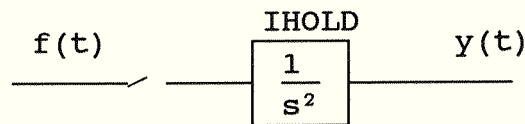


Fig. 4 - Sample-Data System

where "IHOLD" designates a discrete realization process operating on the function $1/s^2$. The separation of these functions has important implications. Hopefully, the reader realizes that the z-transform of a Laplace product is not the product of z-transforms,

$$\mathbf{Z}\{f(s)g(s)\} \neq \mathbf{Z}\{f(s)\}\mathbf{Z}\{g(s)\}$$

When we say that $f(s)$ is the impulse transform $f^*(s)$, producing $f(z)$ and hence $f(kT)$, we must make some assumption about the behavior of the data between the sample points. This is usually done using a data hold function $H(s)$. We may then write,

$$u(z) = f(z)\mathbf{Z}\{H(s)g(s)\}$$

where in this particular case $g(s) = 1/s^2$.

Here we consider four such processes, and one additional process which is the algebraic substitution technique known as both the "bilinear transform" and the "Tustin algorithm". In order to compare the responses to that of the original continuum formulation $y(t)$, note that the "advancing forms" (of the hold function) include one cycle of advance. The complete set of possibilities may then be shown by Fig. 5.

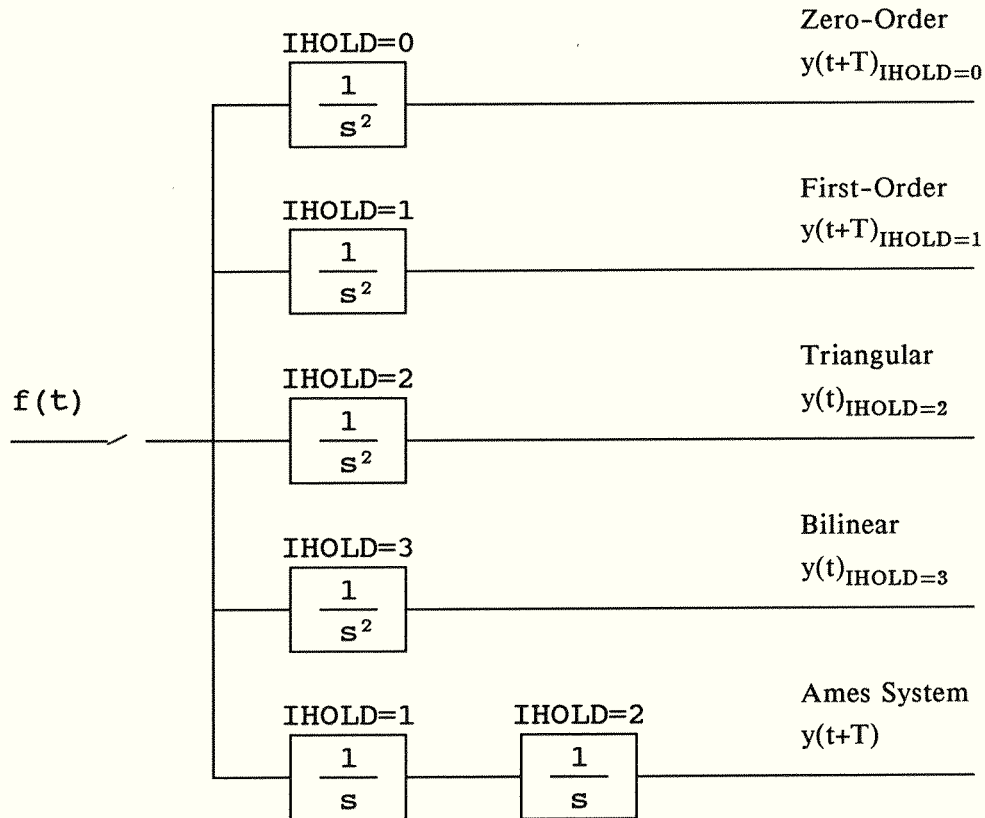


Fig. 5 - Five Discrete Procedures

In Fig. 5 it should be noticed that the Ames System is the only segmented option studied. In this system double integration consists of the Adams-Bashforth algorithm (resultant from the first-order data hold) followed by the trapezoidal algorithm (resultant from the triangular hold). This procedure is not the same as the bilinear algorithm operating upon the double integration process. Although the bilinear algorithm operating on the *single* integration process reduces to the same form as that of the triangular hold algorithm, this is not true for other functions of "s". For example, in z-transform notation the bilinear algorithm operating on double integration is given by,

$$Z\{1/s^2\}_{\text{Bilinear}} = \left(\frac{T^2}{4}\right) \left(\frac{z^2 + 2z + 1}{z^2 - 2z + 1}\right)$$

whereas the triangular algorithm produces,

$$Z\{1/s^2\}_{\text{Triangular}} = \left(\frac{T^2}{6}\right) \left(\frac{z^2 + 4z + 1}{z^2 - 2z + 1}\right)$$

and the combination of Adams-Bashforth and triangular hold algorithms produces,

$$Z\{1/s^2\}_{\text{Ames}} = \left(\frac{T^2}{4}\right) \left(\frac{3z^2 + 2z - 1}{z(z^2 - 2z + 1)}\right)$$

The bilinear algorithm and the triangular hold solution are seen to be different for double integration. The Ames system is seen to be an "advancing form".

For the double integration problem a call to XFRSET (it does not use IHOLD) will produce the numerical equivalent to the following transition matrix,

$$\begin{bmatrix} \Gamma \end{bmatrix} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}$$

and the numerical equivalent to the following forcing function vector,

$$\begin{bmatrix} \Lambda \end{bmatrix} = \begin{bmatrix} T^3/6 \\ T^2/2 \\ T \end{bmatrix}$$

In order to show how these matrix equations may be reduced to scalar sets, the zero-order hold is developed here in detail. By using the transition equations of XFRRUN, the zero-order hold (IHOLD = 0) case produces outputs according to,

$$y_{k+1} = (x_1)_{k+1} = (x_1)_k + T(x_2)_k + \frac{1}{2}T^2 u_k$$

$$y_{k+1} = (x_2)_{k+1} = (x_2)_k + T u_k$$

which may be written in z-transforms by,

$$(z - 1)y(z) = Ty(z) + \frac{1}{2}T^2u(z)$$

$$(z - 1)\dot{y}(z) = Tu(z)$$

or, by breaking the couple, the "advancing forms" of the z-transforms are produced:

$$\frac{y(z)}{u(z)} = \frac{T^2(z + 1)}{2(z - 1)^2}$$

$$\frac{\dot{y}(z)}{u(z)} = \frac{T}{z - 1}$$

These z-transforms are not in concurrent form. In order to put them in concurrent form the highest z-power of the output is solved for, by setting the highest z-power of the input to zero. In this case these operations produce,

$$\frac{zy(z)}{u(z)} = \frac{T^2(1 + z^{-1})}{2(1 - z^{-1})^2}$$

$$\frac{\dot{zy}(z)}{u(z)} = \frac{T}{1 - z^{-1}}$$

The zero-order data hold, the first-order data hold and the Ames system display this shift in their concurrent z-transforms, whereas both the triangular and bilinear forms are already concurrent and thus do not display any shifts. These operations produce Fig. 6, which shows that each of these five discrete realization techniques produce different notation for double integration.

It should be noted that code can be created directly from z-transforms in concurrent form. For example, for double integration the zero-order hold case of Fig. 6 results directly in the FORTRAN code,

$$\begin{aligned}
Y &= 2*YP - YPP + (T**2/2)*(U + UP) \\
YPP &= YP \\
YP &= Y \\
UP &= U
\end{aligned}$$

Note that if both Y and YD (the output and its derivative) are required, the code is even simpler in this case;

$$\begin{aligned}
Y &= Y + T*YD + (T**2/2)*U \\
YD &= YD + T*U
\end{aligned}$$

where these equations are procedural.

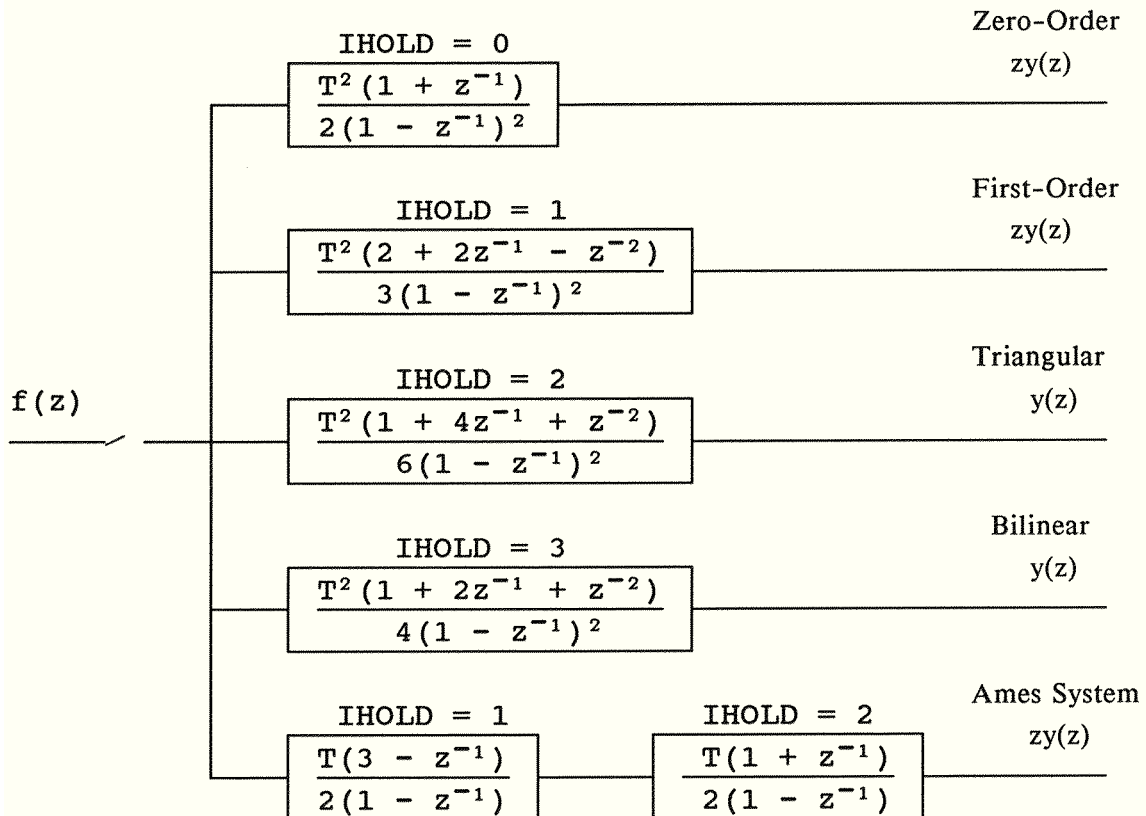


Fig. 6 - Five Discrete (Concurrent) Realizations

Now, since $f(t)$ has been defined above by a trigonometric function, we may explore the time behavior of these algorithms. Both the first integration and the total (double) integration results are shown, in response to the baseline, $f(t)$. Since this is an open-loop system, it should be appreciated that errors are amplified well beyond what would be experienced in closed-loop flight simulation.

With the input "U" being the expression for $f(t)$ given above, the triangular hold solution and the bilinear solution are directly compared with the baseline solution $y(t)$, also given above. The advancing solutions, however, may only be compared to the advanced output, namely $y(t+T)$. Alternately (and this is the selected technique for these graphs), the previous value of the output "Y" is compared to $y(t)$. The three advancing solutions are the zero-order hold, the first-order hold, and the Ames system (advancing for the first integration only).

The time series data $y(t)$ are presented in Figs. 7, 8 and 9 by the solid lines. This system assumes that the forcing function has an undamped natural frequency of 2π , and a damping factor of 0.1. The input $f(t)$, which is the double derivative of $y(t)$, is shown, as well as its two integrals. These curves are repeated on each of the three figures. In order to demonstrate errors, the large cycle time of 100 msec was used (10 samples per cycle of phenomenon).

In Figs. 7(a) through 7(c) the zero-order data hold formulation is shown. The individual data points are shown. The stair-step acceleration behavior is seen to produce a velocity bias in Fig. 7(b). Similarly, the double integral is divergent, as shown in Fig. 7(c).

In Figs. 7(d) through 7(f) the first-order data hold formulation is shown. The extrapolation is shown in Fig. 7(d), and its consequences are shown in Fig. 7(e) for single integration, and in 7(f) for double integration.

The advancing forms of Fig. 7 involve extrapolation, and for an open-loop system cannot hope to produce a very good response. However, if an advance is absolutely required, the first-order data hold is clearly superior to the zero-order data hold.

In contrast, the concurrent forms of Fig. 8 produce a much better response. For the triangular hold of Figs. 8(a) through 8(c), this occurs because interpolation is used.

For the bilinear transform of Figs. 8(d) through 8(f), the results are seen to be similar to the triangular hold (although this cannot be said for general transfer functions).

In Fig. 9(a) through 9(c) the first-order data hold is repeated from Figs. 7(a) through 7(c). It is shown in comparison with the Ames system of Figs. 9(d) through 9(f), where the Adams' algorithm is followed by the trapezoidal algorithm. The velocity data is identical. However, the position data is slightly improved due to the fact that the second integrator uses the triangular data hold.

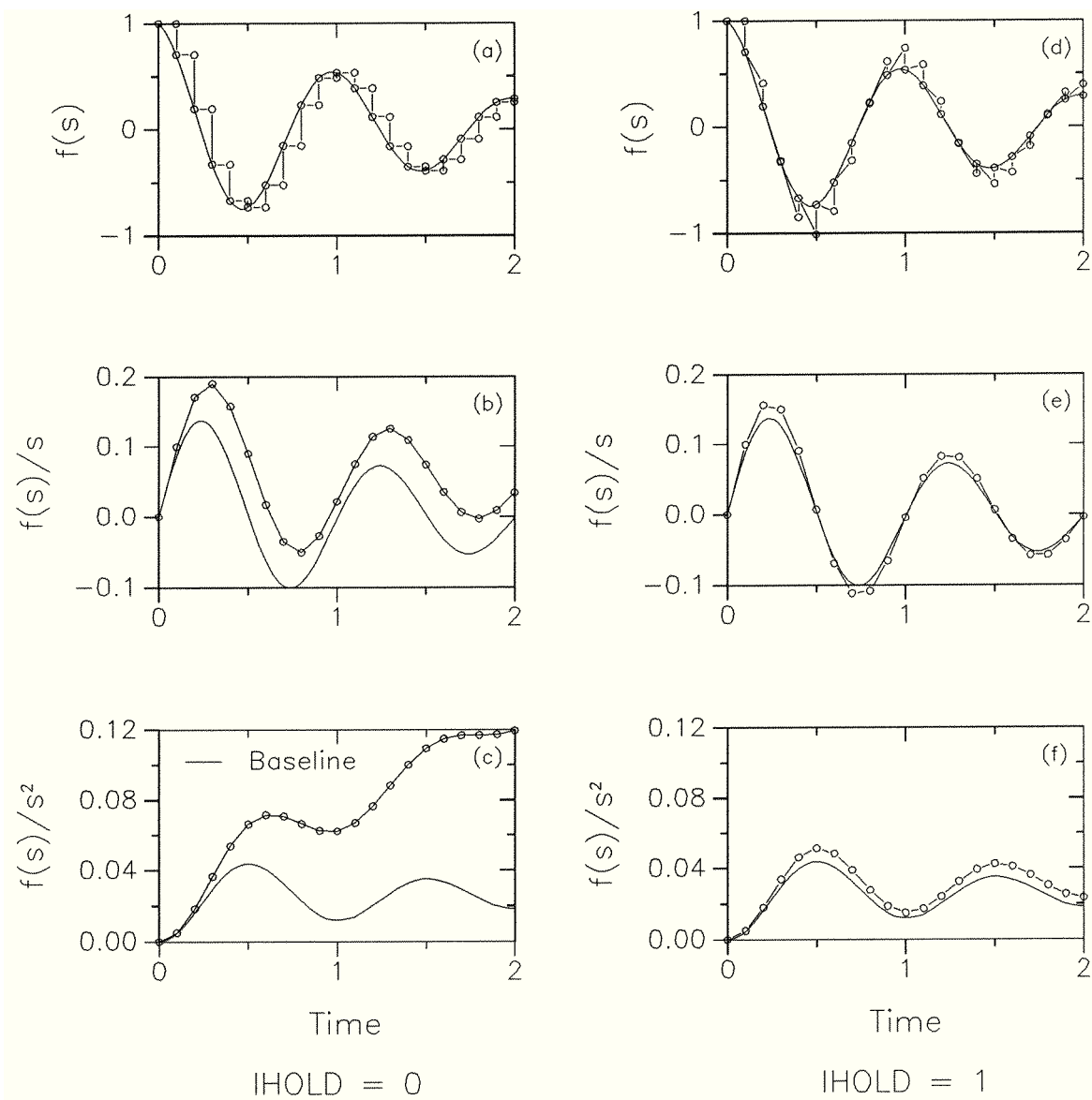


Fig. 7 – Advancing Holds, Zero and First Order

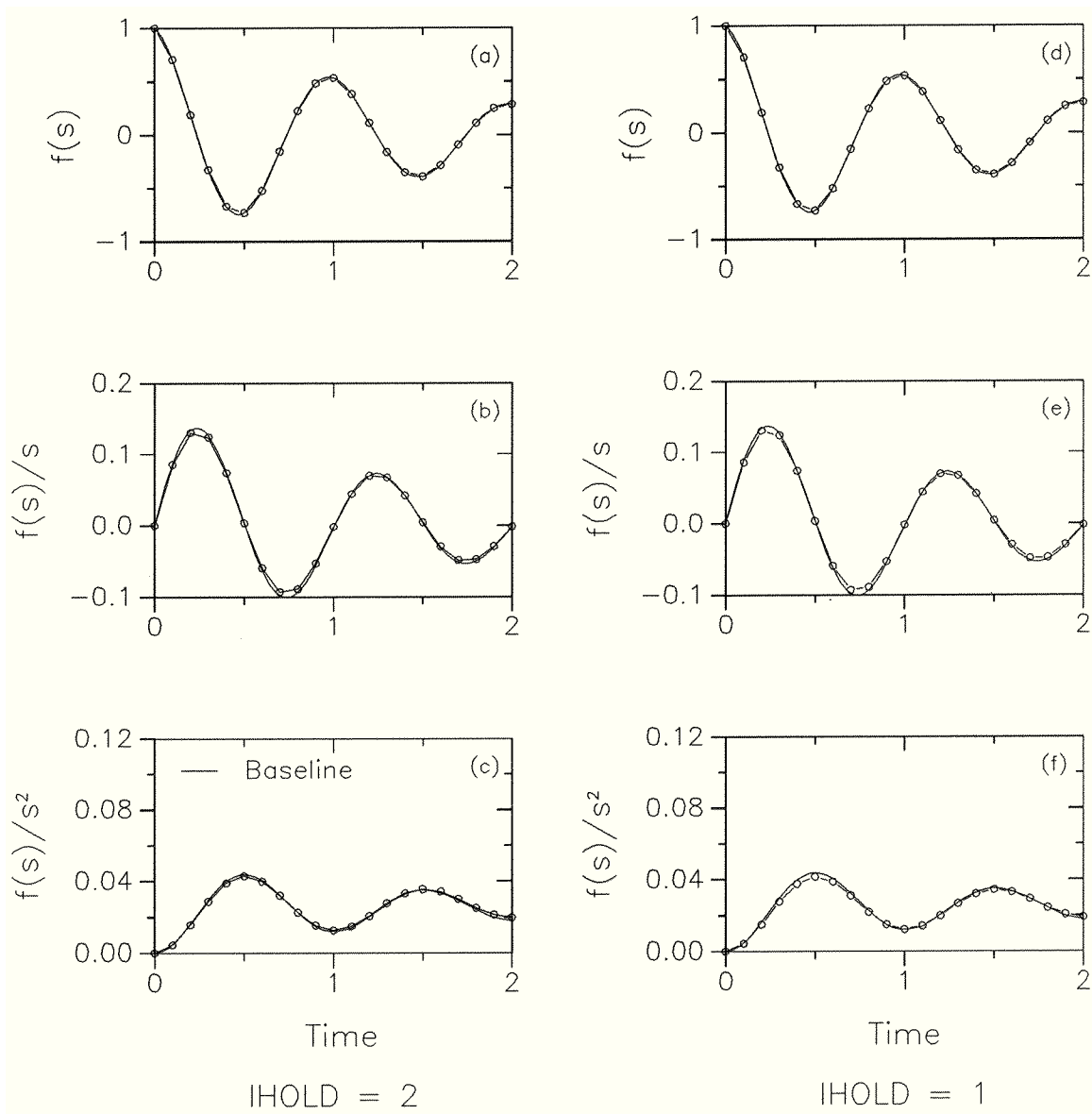


Fig. 8 — Concurrent Holds, Triangular and Bilinear

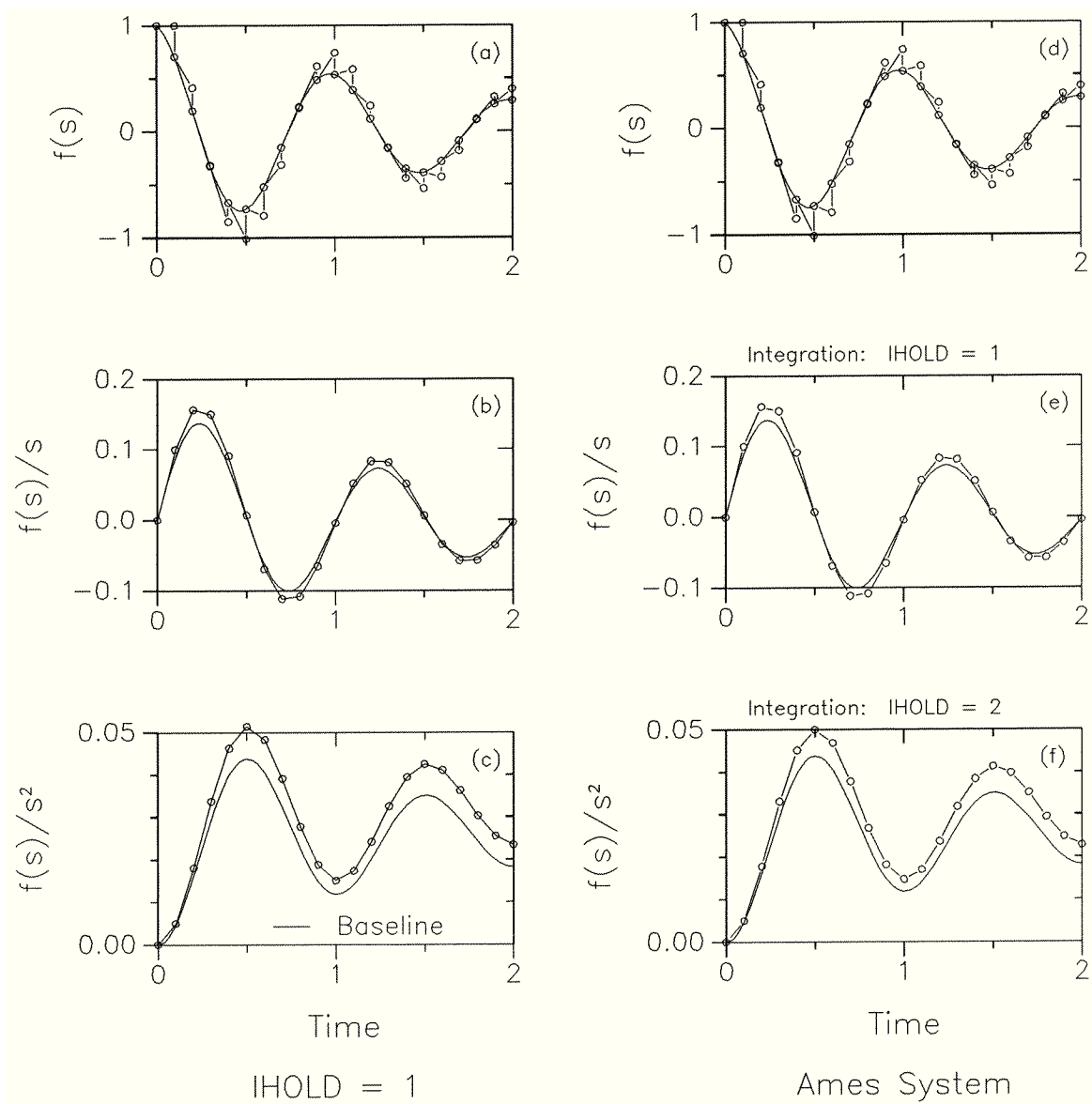


Fig. 9 – Advancing Systems, First Order and Ames

```

C      TITLE                      XFRRUN
C
C      SUBROUTINE XFRRUN(IMODE,IHOLD,DT,N,A,M,B,SETB,RUNB,X,Y)
C
C      R. E. MCFARLAND - NASA - AUGUST 1989
C      VERSION 1.0, NOV. 1, 1990.
C
C      THIS SUBROUTINE WILL HANDLE THE REAL-TIME TRANSITION OF LARGE-ORDER
C      RATIOS OF LAPLACE TRANSFORM POLYNOMIALS.
C
C      XFRSET AND XFRRUN ARE SIMILAR TO THE PROGRAMS FACT AND UPDATE.
C      THE MAJOR IMPROVEMENTS ARE THAT THREE DIFFERENT DATA HOLDS
C      ARE NOW POSSIBLE:
C
C      IHOLD = 0 FOR THE ZERO ORDER HOLD,
C      IHOLD = 1 FOR THE FIRST ORDER HOLD, AND
C      IHOLD = 2 FOR THE TRIANGULAR (TRAPEZOIDAL) HOLD.
C
C      IN ADDITION, AS A SERVICE TO TUSTIN USERS (UGH),
C      IHOLD = 3 FOR THE BILINEAR TRANSFORM SOLUTION
C      (IN THIS CASE SETB IS NOT USED, HENCE,
C      XFRSET NEED NEVER BE CALLED).
C
C      THE OUTPUT FOR THE ZERO OR FIRST ORDER HOLDS ARE ADVANCED ONE CYCLE.
C      THE OUTPUT FOR THE TRIANGULAR HOLD IS CONCURRENT WITH YOUR INPUT.
C      THE OUTPUT FOR THE BILINEAR SOLUTION IS ALSO CONCURRENT.
C
C      ALSO, XFRSET AND XFRBOD MAY BE USED TO GENERATE BODE
C      PLOT DATA FROM YOUR RESULTANT DISCRETE TRANSFER FUNCTIONS.
C      (NOT A REAL-TIME OPERATION).
C      THESE OUTPUTS MAY BE USED FOR A DIRECT COMPARISON
C      TO AN S-PLANE ANALYSIS OF THE ORIGINAL TRANSFER FUNCTION.
C
C      BOTH XFRSET AND XFRRUN USE SUBROUTINE BTYPE FOR ERROR MESSAGES.
C
C      THE BTYPE I/O ROUTINE IS COMPUTER SPECIFIC FOR REAL-TIME
C      SIMULATIONS. A COMPUTER-PORTABLE VERSION (CONSISTING
C      OF ONLY A COUPLE OF LINES OF CODE) IS ALSO AVAILABLE
C      IN STRIKE ACCOUNTS (THE SIMPLE CODE SHOWN IN EXAMPLE AT END).
C
C      DURING MODEL CHECKOUT THE BODE PLOT CAPABILITY USING
C      XFRBOD (AND SUBROUTINE CINVRT) IS INVALUABLE!
C
C      ***** FOR REAL-TIME OPERATIONS (TRANSITION) ***
C
C      WHENEVER YOUR DENOMINATOR COEFFICIENTS CHANGE SUFFICIENTLY
C      ENOUGH THAT YOU WISH TO COMPUTE A NEW TRANSITION MATRIX
C      THEN CALL XFRSET AGAIN. (EXCEPT WHEN IHOLD=3)
C
C      NOTE THAT THIS PERMITS A LARGE NUMBER OF NONSTATIONARY

```

```

C   TRANSFER FUNCTIONS TO TIME-SHARE COMPUTER WORKLOAD.
C
C   NO HISTORY IS STORED IN SETB, ONLY THE FORCING FUNCTION
C   VECTOR AND THE TRANSITION MATRIX.
C   THESE DO NOT CHANGE WITH EITHER THE INPUT (X) HISTORY,
C   OR THE NUMERATOR COEFFICIENTS B(I).
C   SETB IS IDENTIFIED ONLY WITH THE DENOMINATOR POLYNOMIAL.
C
C   WHEN IMODE.GT.0
C   THIS ROUTINE TRANSITIONS THE STATE (INCLUDING NUMERATOR TERMS)
C   IN OPERATE MODE, JUST LIKE SUBROUTINE UPDATE USED TO DO.
C
C   BASED UPON SUBROUTINES FACT AND UPDATE BY:
C
C   R. E. MCFARLAND AND A. B. ROCHKIND
C
C   SEE: "ON OPTIMIZING COMPUTATIONS FOR TRANSITION MATRICES"
C   IEEE TRANSACTIONS ON AUTOMATIC CONTROL, VOL. AC-23,
C   NO. 3, JUNE 1978.
C
C   AND: "FSE PROGRAM SUMMARY DOCUMENT #4", XFFSET, OCT. 1990
C
C   AND: "FSE PROGRAM SUMMARY DOCUMENT #5", XFFRUN, NOV. 1990
C
C   PROVIDING XFRSET HAS BEEN CALLED, XFFRUN SOLVES, IN REAL TIME:
C
C   *****
C   *
C   *   Y   B(M+1)*S**M + B(M)*S**(M-1) + ... + B(2)*S + B(1)   *
C   *   --- = ----- *
C   *   X   S**N + A(N)*S**(N-1) + ... + A(2)*S + A(1)         *
C   *
C   *****
C
C
C
C   IT DOES THIS BY FIRST SOLVING THE TRANSITION PROBLEM,
C
C   *****
C   *
C   *   U           1           *
C   *   --- = ----- *
C   *   X   S**N + A(N)*S**(N-1) + ... + A(2)*S + A(1)         *
C   *
C   *****
C
C
C   WHICH GAINS N U'S, I.E. U, UDOT, UDOUBLEDOT, UP TO UNMINUSONEDOT.
C
C   IF M.EQ.N, UNDOT IS OBTAINED BY THE LINEAR COMBINATION:
C   UNDOT = X - A(N)*UNMINUSONEDOT - ... - A(1)*U

```

```

C BUT X IS ONLY KNOWN AT THE INPUT TIME.  HENCE, UNDOT HAS A MIXED
C TEMPORAL INDEX IF EITHER THE ZERO OR FIRST ORDER HOLDS ARE USED.
C FOR THIS REASON ONLY THE TRIANGULAR HOLD (CONCURRENT OUTPUT) IS
C RECOMMENDED FOR EQUAL-ORDER SYSTEMS (M = N).  (SEE NOTES AT END,
C FOR APPROXIMATIONS EMPLOYED).
C
C IF IHOLD = 3 ALL COEFFICIENTS ARE ASSUMED NONSTATIONARY.
C
C
C *****
C * M CAN BE NO GREATER THAN N *
C *****
C
C
C PERFORMS THE STATE TRANSITION OPERATION IN OPERATE MODE (IMODE.GT.0).
C IN I.C. MODE,
C THE D.C. GAIN IS B(1)/A(1) IF A(1).NE.0 (OTHERWISE 0).
C NOTE, HOWEVER, THAT BY OVERRIDING RUNB VALUES, ANY
C MILDLY CLEVER PROGRAMMER CAN ESTABLISH ARBITRARY
C INITIAL CONDITIONS ON ANY OF THE INTEGRATORS IN THE
C SYSTEM BY USE OF APPROPRIATE I.C.-ONLY STATEMENTS
C (IMODE.LT.0) AFTER THE CALL TO XFRRUN.
C
C
C DEFINITIONS *****
C
C (DIMENSIONS REQUIRED ARE 'AT LEAST' DIMENSIONS)
C
C FOR REAL-TIME, OR TIME TRANSITION, IMODE IS USED:
C
C IMODE -- NEGATIVE = I.C. MODE, POSITIVE = OPERATE MODE.
C
C IHOLD -- 0 = ZERO ORDER HOLD (ADVANCED OUTPUT)
C 1 = FIRST ORDER HOLD (ADVANCED OUTPUT)
C 2 = TRIANGULAR HOLD (CONCURRENT OUTPUT)
C 3 = BILINEAR SOLUTION (CONCURRENT OUTPUT)
C
C DT ----- THE INTERVAL OF TRANSITION (SEC).
C
C N ----- THE ORDER OF THE DENOMINATOR.  MUST BE FROM 1 TO 20.
C
C A --- THE DENOMINATOR COEFFICIENT VECTOR.  MUST HAVE A DIMENSION
C OF AT LEAST N.  (A(N+1) IS ASSUMED UNITY. )
C
C -----
C
C M ----- THE ORDER OF THE NUMERATOR.  MUST BE FROM 0 TO 20.  BUT NOT
C GREATER THAN N.
C
C B --- THE NUMERATOR COEFFICIENTS.  MUST BE DIMENSIONED
C AT LEAST M+1.  B(M+1) IS NOT ASSUMED UNITY.
C
C -----

```

```

C
C SETB - THE BUFFER ASSOCIATED WITH THE DENOMINATOR TRANSFER FUNCTION.
C      THIS BUFFER MUST HAVE THE DIMENSIONS OF AT LEAST
C       $N^2 + N + 1$ 
C
C Y ----- THE SCALAR OUTPUT.
C
C X ----- THE SCALAR INPUT.
C
C RUNB - ANOTHER BUFFER, CONTAINS THE PAST VALUE OF THE INPUT AND THE
C      CURRENT STATES. THIS BUFFER MUST BE UNIQUE FOR EACH X DATA
C      HISTORY, AND BE DIMENSIONED AT LEAST  $N + 1$ .
C
C      RUNB KEEPS TRACK OF THE STATE TRANSITION PROCESS, IN
C      ACCORDANCE WITH UNIQUE INPUT X DATA HISTORY.
C      HENCE, EVEN WITH IDENTICAL DENOMINATORS, DIFFERENT RUNB
C      BUFFERS ARE REQUIRED IF THE INPUTS OR IHOLD ARE DIFFERENT.
C
C      THE INDIVIDUAL STATES ARE LINEARLY COMBINED WITH YOUR
C      NUMERATOR COEFFICIENTS HEREIN IN ORDER TO FORM THE OUTPUT.
C
C      IF ONLY THE B(I) ARE DIFFERENT IN 2 TRANSFER FUNCTIONS,
C      AND THEY HAVE THE SAME INPUT X, THEN THEY MAY SHARE THE
C      SAME RUNB AS WELL AS SETB (IHOLD MUST ALSO BE THE SAME).
C
C      NOTE THAT VARIOUS X AND RUNB COMBINATIONS MAY SHARE ONE
C      SETB PROVIDING ONLY THAT THE DENOMINATORS ARE THE SAME.
C
C
C *****
C *****
C
C FUNCTIONS ONLY OF THE DENOMINATOR POLYNOMIAL:
C
C BUFFER DEFINITIONS:      (SIZE =  $N^2 + N + 1$ )
C
C SETB(1)      C1      FIRST ELEMENT, FORCING FUNCTION VECTOR
C SETB(2)      C2      SECOND ELEMENT, FORCING FUNCTION VECTOR
C ...          ...      ...
C SETB(N+1)    CN+1    N + 1 ELEMENT, FORCING FUNCTION VECTOR
C
C SETB(N+2)    F(1,1)  TRANSITION MATRIX
C SETB(N+3)    F(1,2)  TRANSITION MATRIX
C ...          ...      ...
C SETB(I+1+J*N) F(J,I) TRANSITION MATRIX
C ...          ...      ...
C SETB( $N^2+N+1$ ) F(N,N) TRANSITION MATRIX
C
C
C FUNCTIONS OF SETB, IHOLD, AND THE INPUT X:
C

```

```

C VECTOR DEFINITIONS:      (SIZE = N + 1)
C
C RUNB(1)      U1      FIRST STATE
C RUNB(2)      U2      2ND STATE (1ST DERIV)
C ...          ...      ...
C RUNB(N)      UN      NTH STATE (N-1 DERIV)
C RUNB(N+1)    XK-1    PAST VALUE OF INPUT X
C
C
C *****
C CLASSICAL STATE SPACE SOLUTION TECHNIQUES
C *****
C
C -----
C THE TRANSITION PROBLEM IS SOLVED BY:
C -----
C
C FOR A ZERO-ORDER HOLD (IHOLD = 0), THE ADVANCED STATES ARE:
C
C |U(1)|      |U(1)|  | C(2) |
C |U(2)|      |U(2)|  | C(3) |
C | . | = |F| | . | + | . |X(K)
C | . |      | . |  | . |
C |U(N)|K+1    |U(N)|K  |C(N+1)|
C
C
C FOR A FIRST-ORDER HOLD (IHOLD = 1), THE ADVANCED STATES ARE:
C
C |U(1)|      |U(1)|  | C(2) |  |C(1)|
C |U(2)|      |U(2)|  | C(3) |  |C(2)|
C | . | = |F| | . | + | . |X(K) + | . |[X(K)-X(K-1)]/T
C | . |      | . |  | . |  | . |
C |U(N)|K+1    |U(N)|K  |C(N+1)|  |C(N)|
C
C
C FOR A TRIANGULAR HOLD (IHOLD = 2), THE CONCURRENT STATES ARE:
C
C |U(1)|      |U(1)|  | C(2) |  |C(1)|
C |U(2)|      |U(2)|  | C(3) |  |C(2)|
C | . | = |F| | . | + | . |X(K-1) + | . |[X(K)-X(K-1)]/T
C | . |      | . |  | . |  | . |
C |U(N)|K      |U(N)|K-1 |C(N+1)|  |C(N)|
C
C
C FORM FORCE-AND-MOMENT-PROPORTIONAL QUANTITIES WITH TIME
C APPLICABILITY CONCURRENT WITH THE PILOT INPUT POINT T(K).
C LEAVE THE TRANSITION OF FORCES AND MOMENTS TO VELOCITIES AND
C POSITIONS APPLICABLE AT TIME T(K+1) UP TO STRIKE (OR SMART).
C
C -----
C THE LINEAR COMBINATION PROBLEM IS SOLVED BY:

```

```

C -----
C
C  $Y = B(M+1)*U(M+1) + B(M)*U(M) + \dots + B(1)*U(1)$ 
C
C AND THE ONLY PROBLEM OCCURS WHEN  $M = N$ , IN WHICH CASE
C  $U(M+1) = U(N+1)$  IS NOT AVAILABLE FROM THE STATE VECTOR.
C
C THIS IS NO PROBLEM WHEN THE OUTPUT IS CONCURRENT WITH THE INPUT.
C THIS QUANTITY IS AVAILABLE FROM A CONSIDERATION OF THE DENOMINATOR
C TRANSFER FUNCTION:
C
C  $U(N+1) = X - A(N)*U(N) - A(N-1)*U(N-1) - \dots - A(1)*U(1)$ 
C
C WHERE EVERYTHING IN THIS EQUATION APPLIES AT TIME K, I.E.,
C
C  $(S^{**N})U(K) = X(K) - A(N)*(S^{**(N-1)})U(K) - \dots - A(1)*U(K)$ 
C
C WHEN THE OUTPUT IS ADVANCED, K (ABOVE EQN) GOES TO K+1, AND X(K+1)
C IS UNKNOWN. IT CANNOT BE APPROXIMATED FOR GENERAL TRANSFER FUNCTION
C BEHAVIOR. HENCE, IT IS ASSUMED THAT THE VALUE AT THE BEGINNING OF
C THE NEXT CYCLE IS THE SAME AS IT WAS AT THE BEGINNING OF THIS CYCLE.
C IF YOU DESIRE ANOTHER ASSUMPTION, THIS IS POSSIBLE BY REDUCING
C THE ORDER OF THE NUMERATOR BY ONE AND CREATING YOUR OWN DERIVATIVE
C DATA AS THE INPUT.
C
C THE TRIANGULAR HOLD FORMULATION (IHOLD = 2) IS PREFERRED IN ALL
C CASES. IT HAS THE BEST PERFORMANCE CHARACTERISTICS, AS MAY BE
C SEEN FROM BODE PLOT OUTPUTS (USE SUBROUTINE XFRBOD), IT'S THE
C ONLY PROGRAM IN THE LITERATURE THAT ACTUALLY SHOW YOU WHAT YOU
C WILL GET IN REAL TIME SIMULATION.
C
C FOR REAL-TIME CONTROL SYSTEM MODELING AND AIRCRAFT FORCE/MOMENT
C GENERATION, THE OUTPUTS SHOULD BE CONCURRENT WITH THE PILOT INPUT
C TIME. THIS USUALLY DICTATES CONCURRENCY IN THE MODEL, SO THAT
C THE TRIANGULAR HOLD IS A NATURAL. ALSO, NO PROBLEMS WITH EQUAL
C ORDER SYSTEMS.
C
C
C EXTERNAL DIMENSIONS
C   DIMENSION A(1),SETB(1),B(1),RUNB(1)
C INTERNAL DIMENSIONS
C   DIMENSION YPRIME(20),AK(20)
C
C   *****
C   * EXECUTABLE CODE *
C   *****
C
C   NP1 = N + 1
C
C   IF(IMODE.GT.0) GO TO 270
C

```

```

C          *****
C          * I.C. MODE *
C          *****
C
C TRANSFER FUNCTION RESTRICTIONS
C THESE CHECKS REQUIRED HERE BECAUSE XFRSET DOES NOT CARE WHAT THE
C NUMERATOR IS:
      IF(M.GT.N) THEN
        CALL BTYPE(36,' XFRRUN: NUMERATOR ORDER TOO LARGE. ')
        STOP
      END IF
      IF(M.LT.0) THEN
        CALL BTYPE(36,' XFRRUN: NUMERATOR ORDER TOO SMALL. ')
        STOP
      END IF
C
      IF(N.EQ.0) THEN
        IF(A(1).NE.0.0) THEN
          Y = X*B(1)/A(1)
        ELSE
          Y = 0.0
        END IF
        RETURN
      END IF
C
C I.C. OPERATIONS, ZERO STATE DERIVATIVES.
C
      DO 250 I=1,N
        250 RUNB(I) = 0.0
C
C CHECK FOR EQUAL ORDER SYSTEMS WITH ALL A(I)=0 (EXCEPT N+1)
      IF(M.NE.N) GO TO 256
      DO 254 I=1,N
        IF(A(I).NE.0.0) GO TO 256
      254 CONTINUE
      Y = B(NP1)*X
      GO TO 265
C
      256 CONTINUE
C
      IF(A(1).NE.0.0) THEN
        RUNB(1) = X/A(1)
      END IF
C
      260 Y = B(1)*RUNB(1)
      265 CONTINUE
C SAVE PAST VALUE OF INPUT FOR THIS INPUT/NUMERATOR COMBINATION.
      RUNB(NP1) = X
      IF(IHOLD.LE.2) RETURN
C THE CELL USED FOR PAST VALUE OF HIGHEST STATE DERIVATIVE
C WHEN THE BILINEAR TRANSFORM USED.

```

```

    RUNB(NP1) = X - A(1)*RUNB(1)
    RETURN
C
C          *****
C          * OPERATE MODE *
C          *****
270 CONTINUE
C BILINEAR JUMP IN OPERATE MODE
    IF(IHOLD.GT.2) GO TO 500
C
C RUNB(N + 1) IS USED FOR DEPOSIT OF PREVIOUS INPUT.
    XPREV = RUNB(NP1)
    IF(IHOLD.GT.0) THEN
        DEP = (X - XPREV)/DT
    ELSE
        DEP = 0.0
    END IF
C
C TRANSITION
    DO 290 I=1,N
        SUM = 0.0
        IP1NP2 = N*I + 1
        DO 280 J=1,N
            SUM = SUM + SETB(IP1NP2 + J)*RUNB(J)
280    CONTINUE
290    YPRIME(I) = SUM
C
C BUT THE CURRENT INPUT IS USED AS XPREV FOR ZERO AND FIRST ORDER HOLDS.
C NOTE THAT THE CURRENT INPUT IS IN DEP FOR THE TRIANGULAR HOLD.
    IF(IHOLD.LT.2) XPREV = X
C
C COMBINATION OF TRANSITION TERMS WITH FORCING FUNCTION TERMS.
    DO 300 I=1,N
300    RUNB(I) = YPRIME(I) + SETB(I+1)*XPREV + SETB(I)*DEP
C
    RUNB(NP1) = X
    IF(M.NE.N) GO TO 370
C
C FOR M.LT.N EVERYTHING IS GREAT. GOODBY.
C
C AND EVEN IF M.EQ.N,
C EVERYTHING IS ALSO GREAT UP TO HERE (IN ACCORDANCE WITH
C CLASSICAL THEORY). HOWEVER, FOR EQUAL ORDER SYSTEMS,
C IN ORDER TO COMBINE THE NUMERATOR TERMS, WE
C NEED THE NTH STATE DERIVATIVE (WHEN M = N).
C THIS IS GREAT FOR THE TRIANGULAR DATA HOLD, WHERE
C ALL TERMS ARE CONCURRENT SO THAT THE S**N TERM IS
C AVAILABLE AT K FOR NUMERATOR COMBINATIONS. THE PERTINENT
C LINEAR-COMBINATION EQUATION IS:
C
C  $(S^{*N})U(K) = X(K) - A(N)*(S^{*(N-1)})U(K) - \dots - A(1)*U(K)$ 

```

```

C
C HOWEVER, FOR BOTH THE ZERO AND FIRST ORDER SYSTEMS, WHEN
C M=N, AN ERROR IS INTRODUCED BELOW BECAUSE AN ASSUMPTION
C MUST BE MADE ABOUT WHAT THE INPUT WILL BE ON THE NEXT CYCLE!
C (REPLACE K ABOVE WITH K+1 AND DISCOVER THAT X(K+1) REQUIRED).
C
C FOR THE ZERO-ORDER AND FIRST-ORDER DATA HOLDS, THE BEST ASSUMPTION
C FOR ARBITRARY TRANSFER FUNCTIONS IS THAT THE INPUT VALUE REMAINS
C CONSTANT. SPECIFIC CASES MAY CALL FOR DIFFERENT ASSUMPTIONS, AND
C THESE ARE POSSIBLE WITH A LITTLE EXTERNAL CODING FROM YOU...
C AT LEAST FOR THE ZERO ORDER HOLD CASE, THE BEST ASSUMPTION FOR
C RANDOM INPUTS IS THAT  $X(K+1) = X(K)$ .
C
C CONSULT SUBROUTINE XFRBOD'S OUTPUTS TO DETERMINE THE CHARACTERISTICS
C OF YOUR TRANSFER-FUNCTION/DATA-HOLD REQUIREMENTS.
C
C FOR THE ABOVE REASONS THE ZERO ORDER AND FIRST ORDER DATA
C HOLD SYSTEMS ARE NOT RECOMMENDED FOR EQUAL ORDER SYSTEMS (M=N).
C
C AS A POSSIBLE ALTERNATIVE:
C YOU MIGHT, FOR INSTANCE, CREATE YOUR OWN PSEUDO-INPUT AS THE
C DERIVATIVE OF THE REAL INPUT, THEREBY REDUCING M SO THAT IT
C IS LESS THAN N. WHEN YOU DO THIS, YOU WILL CERTAINLY GAIN AN
C APPRECIATION FOR THE PROBLEM OF MATCHING TEMPORAL INDICES.
C
      Y = X
      DO 360 I=1,N
      Y = Y - A(I)*RUNB(I)
360 CONTINUE
C
      Y = B(M+1)*Y
      MP1 = M
      GO TO 380
C
370 Y = 0.0
      MP1 = M + 1
C
380 CONTINUE
C
C ADD IN THE REMAINING NUMERATOR TERMS
      DO 390 I=1,MP1
390 Y = Y + B(I)*RUNB(I)
C
      RETURN
C
C *****
C WHEN IHOLD = 3, A STATE SPACE SOLUTION IS NOT OBTAINED.
C THE BILINEAR TRANSFORM, WHICH IS HERE PRESENTED IN GENERAL
C SOLUTION FORM, IS ALSO CALLED THE TUSTIN METHOD. IT IS AN
C ALGEBRAIC SOLUTION, AND SHOULD NOT BE USED FOR NOTCH FILTERS.
C (FREQUENCY-AXIS DISTORTION OCCURS).

```

```

C
500 CONTINUE
C NONSTATIONARY COEFFICIENTS O.K., AND XFRSET NOT USED.
  XK = 0.5*DT
  ZK = 1.0
  DSUM = 1.0
C
  DO 510 I = 1,N
    AK(I) = ZK
    ZK = ZK*XK
510 DSUM = DSUM + ZK*A(NP1 - I)
C
  ASUM = 0.0
  DO 530 J=1,N
    ESUM = 0.0
    DO 520 NN=J,N
      NNP1 = NN + 1
520  ESUM = ESUM + A(NNP1-J)*(RUNB(NN) + XK*RUNB(NNP1))
530  ASUM = ASUM + AK(J)*ESUM
C
  ASUM = (X - ASUM)/DSUM
  SUMS = ASUM
C
  DO 540 I = N,1,-1
    SUMS = RUNB(I) + XK*(SUMS + RUNB(I+1))
540 YPRIME(I) = SUMS
C
  DO 550 I=1,N
550 RUNB(I) = YPRIME(I)
C
  RUNB(NP1) = ASUM
C
  GO TO 370
C
C END OF CODE.....
C
C *****
C  * SUGGESTION AREA *
C  *****
C
C *****
C SUBROUTINE BTYPE, IF YOU DO NOT HAVE IT IN A NON-REAL-TIME
C APPLICATION (REMOVE COMMENTS)
C  SUBROUTINE BTYPE(NCHARS,CHAR)
C  CHARACTER*1 CHAR(NCHARS)
C  WRITE(*,'(1X,60A1)')(CHAR(I),I=1,NCHARS)
C  RETURN
C  END
C *****
END

```